Dokumentationspunkt 3.1

Dokumenttyp

- Kontaktversuch 12. Februar 2026
- Kontaktversuch an sieben verschiedenen institutionellen Schnittstellen
- Dokumentierter Resonanzstatus: R0/7

Wiederholter Kontaktversuch bei selektierter institutioneller Schnittstelle

Ergänzende fachliche Einordnung – ITER → DEMO Safety-Kernel (

Originaldokument nachfolgend archiviert.

———

Strukturierte Einordnung des Originaldokuments

Sehr geehrte Damen und Herren,

ergänzend zu einer vor einiger Zeit übersandten Andock-Notiz zum Themenfeld ITER → DEMO wurde eine kurze fachliche Präzisierung nachgereicht.

Der Hintergrund dieser Ergänzung war rein struktureller Natur:

Im Zuge weiterer Ausarbeitung zeigte sich, dass die ursprünglich beschriebene Safety-first-Logik nicht nur als konzeptioneller Rahmen, sondern als formalisierbarer Übergangs- und Lernmechanismus lesbar ist – insbesondere im Hinblick auf den Unterschied zwischen experimenteller und betrieblicher Systemlogik.

Kernpunkt der Präzisierung war die explizite Trennung zwischen:

- harten, nicht lernenden Schutzmechanismen (Interlocks / deterministische Grenzen)
- modell- oder datenbasierten Wächterstrukturen (Guardians / Shadow Mode)
- versionierten Meta-Anpassungen (Grenzen / Korridore ausschließlich evidenzbasiert)

Lernen wurde dabei nicht primär als Optimierungs- oder Regelungsproblem beschrieben, sondern als kontrollierter, auditierbarer Prozess innerhalb stabiler Systemgrenzen.

Grenzverschiebungen erfolgen ausschließlich:

- versioniert,
- nachvollziehbar,
- evidenzbasiert,
- und unter dokumentierter Rückfallmöglichkeit.

Die ergänzende Einordnung verstand sich ausdrücklich nicht als technischer Lösungsvorschlag oder Projektangebot, sondern als strukturierende Lesart möglicher Übergänge zwischen experimentellen und betrieblichen Sicherheitslogiken.

Thematische Schwerpunkte:

- Eskalationslogik unter Unsicherheit
- modellbedingte Nichtdeterministik
- Failure-Injection-basierte Validierung
- Trennung von Lernen und Steuerhoheit

Eine formale Erwartung oder Handlungsaufforderung war damit nicht verbunden.

———

Technische Präzisierung (archivierte Kernaussagen)

Die dokumentierte Struktur beschreibt:

- definierte Zustands- und Handlungsräume,
- feste Constraints,
- probabilistische Unsicherheitsmodellierung,
- versionierte Sicherheitskorridore,
- sowie die strikte Trennung zwischen:
  - Modelllernen,
  - operativer Entscheidung,
  - und Meta-Governance.

Zentrale dokumentierte Prinzipien:

- Entscheidungen erfolgen ausschließlich innerhalb definierter Sicherheitsräume.
- Lernen verändert Modellgüte, jedoch nicht automatisch Systemgrenzen.
- Grenzänderungen sind nur evidenzbasiert, versioniert und auditierbar zulässig.
- Online-Betrieb erfolgt ausschließlich innerhalb fixierter Sicherheitsparameter.
- Schleichende Drift und implizite Risikoerhöhung sollen strukturell ausgeschlossen werden.

Dokumentiert wurde außerdem:

- die mathematische Formalisierung zulässiger Zustandsräume,
- probabilistische Sicherheitsbedingungen,
- Bayes-basierte Modellaktualisierung,
- sowie die formale Trennung zwischen:
  - Parameterlernen,
  - Constraints,
  - und Hazard-Grenzen.

———

Archivstatus

Dokument vollständig archiviert.
Originalkommunikation entpersonalisiert und strukturell unverändert dokumentiert.

———

Dokumentierter Zeitverlauf

Zeitraum nach ergänzender Übermittlung

Dokumentierter Status nach Ablauf des vorgesehenen Beobachtungs- und Reflexionszeitraums:

- Keine dokumentierte fachliche Rückmeldung
- Keine dokumentierte Weiterführung
- Keine erkennbare Anschlusskommunikation
- Keine öffentliche Bezugnahme dokumentiert

———

Dokumentarischer Hinweis

Die Archivierung erfolgt chronologisch und zeitversetzt entsprechend der innerhalb des Projekts definierten Dokumentationsmethodik.

Die Darstellung dient ausschließlich der historischen Nachvollziehbarkeit dokumentierter Kommunikations-, Präzisierungs- und Kopplungsprozesse innerhalb technischer, institutioneller und gesellschaftlicher Wahrnehmungsräume.

Originaltext

Ergänzende fachliche Einordnung – ITER → DEMO Safety-Kernel (v0)

Sehr geehrte Frau …,

ergänzend zu meiner vor einiger Zeit übersandten Andock-Notiz zum Themenfeld ITER → DEMO möchte ich eine kurze fachliche Präzisierung nachreichen.

Der Hintergrund dieser Ergänzung ist rein struktureller Natur:
Im Zuge weiterer Ausarbeitung hat sich gezeigt, dass die ursprünglich beschriebene Safety-first-Logik nicht nur als konzeptioneller Rahmen, sondern als formalisierbarer Übergangs- und Lernmechanismus lesbar ist – insbesondere im Hinblick auf den bekannten Unterschied zwischen experimenteller und betrieblicher Systemlogik.

Kernpunkt dieser Präzisierung ist die explizite Trennung von:

• harten, nicht lernenden Schutzmechanismen (Interlocks / deterministische Grenzen)
• modell- oder datenbasierten Wächterstrukturen (Guardians / Shadow Mode)
• versionierten Meta-Anpassungen (Grenzen / Korridore ausschließlich evidenzbasiert)

Damit wird Lernen nicht als Regelungs- oder Optimierungsproblem behandelt, sondern als kontrollierter, auditierbarer Prozess innerhalb stabiler Systemgrenzen.
Grenzverschiebungen erfolgen nicht online oder heuristisch, sondern ausschließlich über nachvollziehbare Evidence (Tests, Logs, Nachanalysen).

Diese Sichtweise versteht sich weiterhin ausdrücklich nicht als technische Lösung oder Projektvorschlag, sondern als mögliche strukturierende Lesart für den ITER → DEMO-Übergang, insbesondere für Fragestellungen wie:

• Eskalationslogik unter Unsicherheit
• Umgang mit modellbedingter Nichtdeterministik
• Failure-Injection-basierte Validierung
• Trennung von Lernen und Steuerhoheit

Sollte die Unterlage fachlich an anderer Stelle besser anschlussfähig sein, ist sie selbstverständlich unverändert weitergebbar.
Eine formale Erwartung oder Handlungsaufforderung ist damit nicht verbunden.

Vielen Dank für die Zeit und die fachliche Einordnung.

Freundliche Grüße
Werner Nicolai

Technische Präzisierung – Safety-Kernel & Learning-Logik (v0)

Technische Einordnung (strukturorientiert, normneutral)

Die zugrunde liegende Logik lässt sich formal als kontrollierte Bewegung innerhalb eines zulässigen Zustands- und Handlungsraums beschreiben.

Ein Systemzustand sei

$x \in \mathcal{X} \subset \mathbb{R}^n$

mit Handlungen / Stellgrößen

$u \in \mathcal{U} \subset \mathbb{R}^m$

Randbedingungen (physikalisch, technisch oder systemisch) werden durch Constraints beschrieben:

$g(x, u) \leq 0$

Der zulässige Betriebsbereich ergibt sich damit als:

$\mathcal{S} = \{ (x, u) \mid g(x, u) \leq 0 \}$

Handlungen und Entscheidungen finden ausschließlich innerhalb dieser Menge statt.

———

Entscheidungsprinzip (Annäherung statt Wahrheitsfestschreibung)

Ein System verfolgt Zielgrößen über eine Kosten- oder Bewertungsfunktion J(u; x).
Entscheidungen können als Optimierungsproblem unter Nebenbedingungen formuliert werden:

$u^*(x) = \arg\min J(u; x)$
s.t. $(x, u) \in \mathcal{S}$

Unter Unsicherheit wird die harte Randbedingung zweckmäßig probabilistisch gefasst:

$\mathbb{P}(g(x, u) \leq 0) \geq 1 - \varepsilon$

mit ε als explizit gewähltem Restrisiko.

Damit bleibt die Entscheidungslogik deterministisch, während Unsicherheit transparent modelliert wird.

———

Lernen vs. Grenzverschiebung (zentrale Trennung)

Modell- oder datenbasierte Anteile besitzen Parameter θ.
Lernen wird als Wissensaktualisierung behandelt, z. B. über ein Bayes-Update:

$p(\theta \mid D) \propto p(D \mid \theta)\, p(\theta)$

Dieses Update verändert Modellgüte, jedoch nicht automatisch die Systemgrenzen.

Grenzen / Korridore werden als separate Meta-Parameter ϕ beschrieben:

$\mathcal{S}_\phi = \{ (x, u) \mid g_\phi(x, u) \leq 0 \}$

Online-Betrieb erfolgt unter fixiertem ϕ_0.

Eine Anpassung von ϕ erfolgt ausschließlich versioniert und evidenzbasiert:

$\phi_{k+1} = \text{Update}(\phi_k ; E_k)$

wobei E_k explizite Evidence (Tests, Logs, Failure-Injections, Reviews) darstellt.

Eine Grenzänderung ist nur zulässig unter nachweisbarer Hazard-Schranke:

$\mathbb{P}$ (Hazard | ϕ_{k+1}) ≤ δ

———

Systemische Konsequenz

Diese Struktur trennt strikt zwischen:

• Entscheidung innerhalb definierter Grenzen
• Lernen als Modellverbesserung
• Grenzänderung als bewusster Meta-Akt

und verhindert damit schleichende Drift, implizite Risikoerhöhung oder nicht nachvollziehbare Systemänderungen.

———

Kernprinzip (kompakt)

Entscheidungen optimieren innerhalb stabiler Constraints,
Lernen verbessert Modelle ohne Steuerhoheit über Grenzen,
Grenzänderungen erfolgen ausschließlich kontrolliert, versioniert und evidenzbasiert.

Gesendet mit der WEB.DE Mail App

Am 17.12.25 um 16:26 schrieb aw.discovery@web.de

>
> Sehr geehrte Frau Pisselhoy,
> ich sende Ihnen im Anhang eine kurze fachliche Andock-Notiz zum
> Themenfeld ITER → DEMO.
> Die Unterlage ist kein fertiger Vorschlag und kein Auftrag, sondern
> ein offener Arbeitsstand aus laufender Zusammenarbeit. Uns war
> wichtig, die zugrunde liegende Intention (Safety-first, konservativer
> Einstieg, schrittweises Lernen) transparent zu machen, bevor man in
> Details einsteigt.
> Der Workshop-Teil ist bewusst pragmatisch gehalten und zielt
> ausschließlich darauf ab, technische TBD-Felder (Sensoren,
> Grenzwerte, Abtastraten, Interlocks) gemeinsam zu konkretisieren –
> ohne Architektur- oder Optimierungsdebatten.
> Falls das Dokument fachlich bei Kolleginnen oder Kollegen besser
> aufgehoben ist, ist es ausdrücklich weitergebbar. Eine formale
> Erwartung ist damit nicht verbunden.
> Vielen Dank fürs Lesen und Einordnen.
> Freundliche Grüße
>
> Einleitung
>
> Warum dieses Dokument existiert – und wie es gelesen werden sollte
> Dieses Dokument ist kein fertiges Systementwurf, kein Lehrbuch und kein abgeschlossener Plan.
> Es ist eine Arbeitsgrundlage.
> Es entstand aus der Überzeugung, dass komplexe, sicherheitskritische Systeme – insbesondere im

> Kontext von Fusion und DEMO – nicht durch Einzelideen, Genialität oder spätere Optimierung
> tragfähig werden, sondern durch frühe Klarheit, gemeinsame Verantwortung und
nachvollziehbare
> Entscheidungen.
> Was hier vorliegt, ist bewusst ein konservativer Startpunkt.
> Nicht, weil wir wenig ambitioniert wären,
> sondern weil wir gelernt haben, dass Sicherheit, Lernen und Vertrauen eine Reihenfolge haben.
> Ein Quantensprung – ja. Ein Zauberwerk – nein.
> Für viele Leserinnen und Leser wird dieses Dokument in seiner Form und Konsequenz zunächst
> ungewohnt sein.
> Nicht wegen einzelner Konzepte – Interlocks, Redundanz, Zustände, Zeitfenster kennt jede
> Ingenieurdisziplin –
> sondern wegen der konsequenten Art, wie sie hier zusammengeführt werden.
> Dieser „Quantensprung“ ist kein Sprung in die Abstraktion,
> sondern ein Schritt hin zu Explizitheit:
> • Was ist eingefroren – und warum?
> • Wo darf gelernt werden – und wo ausdrücklich nicht?
> • Wie wird aus Beobachtung eine Entscheidung?
> • Wie bleibt alles rekonstruierbar, auch Jahre später?
> Nichts davon ist Magie.
> Alles davon ist handwerklich begründbar.
> Kein Einzelwerk – sondern bewusst offen gelegt
> Dieses Dokument erhebt keinen Anspruch auf Vollständigkeit im operativen Detail.
> Im Gegenteil: Es macht offen sichtbar, wo Wissen noch ergänzt werden muss – und durch wen.
> Das ist Absicht.
> Die hier beschriebene Architektur, der Safety-Kernel, die Failure-Injection-Logik und die
> Workshop-Struktur sind kein Ausdruck individueller Brillanz, sondern ein Gerüst für
> Zusammenarbeit:
> • Ingenieurwissen
> • Betriebserfahrung
> • Safety-Expertise
> • regulatorische Anforderungen
> werden hier nicht ersetzt, sondern eingeladen.
> Wenn dieses Dokument funktioniert, dann nicht, weil es „richtig“ ist,
> sondern weil es gemeinsam richtig gemacht wird.
> Wie dieses Dokument gelesen werden sollte
> Wir empfehlen ausdrücklich:
> • Nicht alles auf einmal lesen zu wollen
> • Nicht nach Optimierung oder Performance zu suchen
> • Nicht nach dem „besten“ Modell oder Tool zu fragen
> Stattdessen:
> • Erst das Prinzip verstehen
> • Dann die Leitplanken akzeptieren
> • Und anschließend gemeinsam die offenen Felder füllen
> Die markierten TBD-Stellen sind kein Mangel.
> Sie sind der Arbeitsauftrag.
> Haltung
> Dieses Dokument folgt einer klaren Haltung:
> Sicherheit vor Leistung.
> Lernen vor Optimierung.
> Nachvollziehbarkeit vor Geschwindigkeit.
> Zusammenarbeit vor Einzelmeinung.
> Wenn wir diese Haltung teilen, dann ist alles Weitere Ingenieurarbeit –
> und genau dafür ist dieses Dokument geschrieben.
> Startstrategie: “Safety-first Learning”
> 1) Zwei-Schichten-Schutz (unverhandelbar)
> Schicht A: Interlocks (hart, 1-of-1)
> • physik-/regelbasierte Grenzen, die sofort auf Hard Abort gehen
> • nicht lernend, nicht verhandelbar
> Schicht B: Guardian-Voting (konservativ, K-of-M)

> • viele unabhängige Wächter, die auf Safe Shutdown eskalieren
> • lernend nur in dem Sinn, dass wir später Schwellen feinjustieren
> Das verhindert “frühe, teure Fehler”, weil du nie in einen Bereich kommst, in dem du erst
> im Nachhinein merkst, dass du zu spät warst.
> ———
> Konservatives Startmodell (konkret)
> 2) Signale
> Jeder Wächter i liefert diskret:
> s_i(t)\in\{OK, WARN, TRIP\}
> 3) Fensterregel (schnell + auditierbar)
> Wähle ein kurzes Zeitfenster W (z. B. 100–300 ms, abhängig vom Prozess) und zähle:
> T(t)=\sum_{i=1}^{M}\mathbf{1}[s_i(t)=TRIP]
> Regel (maximale Sicherheit):
> • Hard Abort, wenn ein Interlock triggert.
> • Safe Shutdown, wenn T(t)\ge K für irgendeinen Zeitpunkt im Fenster.
> Für maximalen Sicherheitsstart gilt typischerweise:
> • M=10
> • K=2 (sehr konservativ)
> oder sogar K=1 nur für ausgewählte “sehr vertrauenswürdige” Guards.
> 4) Diversitätsanforderung (wichtig!)
> Damit nicht zwei identische Sensorfehler alles stoppen, gruppierst du in Klassen \mathcal{C}
> _1,\dots,\mathcal{C}_p (z. B. Physikmodell / Anomalie-ML / Sensor-Plausibilität / Aktor-
Sättigung
> …).
> Konservativ + robust:
> \exists c_1\neq c_2:\ \sum_{i\in\mathcal{C}_{c_1}}\mathbf{1}[TRIP]\ge 1 \ \land\
> \sum_{i\in\mathcal{C}_{c_2}}\mathbf{1}[TRIP]\ge 1
> Also: mindestens zwei Trips aus zwei unterschiedlichen Klassen ⇒ Shutdown.
> Das ist ein riesiger Sicherheitsgewinn gegenüber „2 beliebige Stimmen“.
> ———
> Lernen ohne Risiko: “Shadow Mode” zuerst
> 5) KI/Modelle dürfen am Anfang nicht entscheiden
> Sie laufen parallel (“shadow”), loggen alles, schlagen nur vor.
> Du nutzt sie für:
> • Risikoabschätzung
> • Frühwarnung
> • Nachanalyse
> Aber die Entscheidung bleibt bei:
> • Interlocks
> • Guardian-Quorum
> So lernst du, ohne dich auf unkalibrierte Modelle zu verlassen.
> ———
> 6) Progression: Nur lockern, nie improvisieren
> Du definierst Versionsstufen:
> • v0 (ultra safe): K=2 + Diversität + kurze Fenster, viele Abbrüche erlaubt
> • v1: K bleibt, aber Schwellen der Guards werden besser kalibriert
> • v2: K leicht erhöhen oder Fenster/Hysterese feinjustieren
> • v3: erst dann Optimierung und Performance
> Wichtig: jede Version ist dokumentiert, und Rückfall auf v0 ist immer möglich.
> ———
> 7) Warum das “tragfähig” ist
> Weil du damit:
> • keine frühen Skandale/katastrophalen Fehlschläge riskierst
> • Vertrauen durch Nachvollziehbarkeit aufbaust
> • das System in Ruhe kalibrierst
> • und erst später Effizienz optimierst
> Das ist exakt die Reihenfolge, die tragfähige Projekte über Jahrzehnte rettet.
> v0-SAFETY-KERNEL
> (oberste Ebene

> –
> nicht verhandelbar)
> Ziel von v0 ist Schadensvermeidung vor Leistungsgewinn.
> Kein Optimierungsdruck. Kein Heroismus. Keine Abkürzungen.
> ———
> 1) Grundprinzipien (Systemverfassung)
> Diese Prinzipien sind konstitutiv – sie gelten unabhängig von Technik, Domäne oder späteren
> Versionen.
> 1. Safety > Performance
> Jeder Abbruch ist ein Erfolg, wenn er Schaden verhindert.
> 2. Kein Single Point of Failure
> Weder Mensch noch KI noch Sensor noch Modell dürfen allein entscheiden.
> 3. Entscheidung ≠ Berechnung
> Berechnung darf schnell, komplex und probabilistisch sein.
> Entscheidung ist diskret, begründet und rekonstruierbar.
> 4. Lernen nur im geschützten Raum
> Modelle lernen, ohne Steuerhoheit zu haben (Shadow Mode).
> 5. Rückfall immer möglich
> Jede spätere Version kann jederzeit auf v0 zurückgesetzt werden.
> ———
> 2) Systemebenen (klar getrennt)
> Ebene 0
> —
> Interlocks (hart, absolut)
> • Physik / Regeln / Normen
> • 1-of-1-Logik
> • Sofortiger Hard Abort
> • Kein Lernen, keine KI, keine Diskussion
> Diese Ebene ist nicht Teil des Experiments.
> ———
> Ebene 1
> —
> Safety Kernel (v0, unser Fokus)
> • Mehrere unabhängige Wächter
> • Diskrete Signale
> • Konservatives Quorum
> • Zeitfenster + Diversität
> • Vollständiges Logging
> Hier findet kontrolliertes Lernen ohne Risiko statt.
> ———
> Ebene 2
> —
> Optimierung & KI (nur Vorschläge)
> • Rechnet Wahrscheinlichkeiten
> • Erkennt Muster
> • Gibt Empfehlungen
> • Keine Entscheidungsgewalt
> Diese Ebene darf scheitern, ohne Schaden anzurichten.
> ———
> 3) Wächterarchitektur (v0)
> Anzahl
> M = 10 \quad \text{(bewusst überschaubar, aber redundant)}
> Klassen (Diversität!)
> Wir teilen die 10 Wächter in mindestens 4 Klassen:
> 1. Physik / Regeln (modellbasiert, nicht lernend)
> 2. Anomalie / Abweichung (ML, statistisch)
> 3. Sensor- & Aktor-Plausibilität
> 4. Systemkohärenz / Meta-Checks
> Kein Quorum darf aus nur einer Klasse stammen.
> ———

> 4) Wächter-Signale (maximal auditierbar)
> Jeder Wächter liefert ausschließlich:
> s_i(t) \in \{\textbf{OK},\ \textbf{WARN},\ \textbf{TRIP}\}
> • OK: alles im Rahmen
> • WARN: Auffälligkeit, aber kein Abbruch
> • TRIP: Grenze erreicht → sicherheitsrelevant
> Keine Scores nach außen.
> Keine Wahrscheinlichkeiten in Entscheidungen.
> ———
> 5) Entscheidungslogik (ultra-konservativ)
> Zeitfenster
> Kurzes, festes Fenster:
> W = 100\text{–}300\,\text{ms} \quad (\text{prozessabhängig})
> Quorumregel (v0)
> Hard Abort
> • wenn irgendein Interlock triggert (Ebene 0)
> Safe Shutdown, wenn innerhalb von W:
> • mindestens 2 TRIP
> • aus zwei verschiedenen Klassen
> Formal:
> \exists c_1 \neq c_2:\
> \sum_{i \in \mathcal{C}_{c_1}} \mathbf{1}[TRIP] \ge 1
> \ \land\
> \sum_{i \in \mathcal{C}_{c_2}} \mathbf{1}[TRIP] \ge 1
> Das ist der größte Sicherheitssprung gegenüber einfachen Mehrheiten.
> ———
> 6) Zustandsmaschine (minimal & stabil)
> • M1
> –
> Normal
> • M2
> –
> Warn / Degraded
> • M3
> –
> Safe Shutdown
> • M4
> –
> Hard Abort
> Übergänge:
> • OK → WARN: einzelne WARNs
> • WARN → Safe: Quorum erfüllt
> • Jeder Zustand → Abort: Interlock
> Kein Zurückspringen ohne Haltezeit.
> ———
> 7) Lernen ohne Risiko (Shadow Mode)
> • KI/Modelle laufen parallel
> • Loggen:
> • was sie gesehen hätten
> • wann sie gewarnt hätten
> • Kein Einfluss auf Steuerung
> Nachanalyse:
> • False Positives
> • Missed Warnings
> • Timing
> • Korrelationen
> So entsteht Wissen ohne operative Gefahr.
> ———

> 8) Logging & Nachvollziehbarkeit (Pflicht)
> Jede Modusänderung speichert mindestens:
> • Zeitstempel
> • aktueller Modus → neuer Modus
> • beteiligte Wächter + Klassen
> • Signale s_i
> • aktives Zeitfenster
> • Version des Safety Kernels
> Ohne vollständigen Log kein Fortschritt zur nächsten Stufe.
> ———
> 9) Warum das tragfähig ist
> • Kleine Erfolge zählen (System bleibt heil)
> • Fehler sind informativ, nicht zerstörerisch
> • Vertrauen wächst durch Wiederholbarkeit
> • Öffentlichkeit / Stakeholder können es verstehen
> • Spätere Effizienz ist aufbauend, nicht riskant
> v0
> –
> DIE 10 WÄCHTER (GUARDIAN SET)
> Überblick: Klassen & Zweck
> Wir teilen bewusst in 4 Klassen, damit Diversität real ist:
> Klasse Zweck
> A Physikalisch / regelbasiert (nicht lernend)
> B Systemverhalten / Anomalien
> C Sensorik & Aktorik
> D Meta-Ebene / Systemkohärenz
> 🟥 KLASSE A
> –
> Physik & harte Regeln (Fundament)
> Guardian A1
> –
> Absolute Grenzwerte
> Art: Regelbasiert
> Beobachtet:
> • Max/Min-Werte (Energie, Feldstärke, Temperatur, Druck, etc.)
> TRIP, wenn:
> • irgendein absoluter Grenzwert überschritten wird
> ➡️ Nahe an Interlock, aber noch im Safety Kernel
> ➡️ Sehr hohe Vertrauensstufe
> ———
> Guardian A2
> –
> Art: Regelbasiert
> Beobachtet:
> Dynamische Änderungsraten
> • Ableitungen / Rampen (dX/dt)
> TRIP, wenn:
> • ein Wert sich zu schnell ändert, auch wenn er noch „legal“ ist
> ➡️ Schützt vor instabilen Übergängen
> ➡️ Typischer Frühindikator
> ———
> Guardian A3
> –
> Energiebilanz / Konsistenz
> Art: Modellbasiert (nicht lernend)
> Beobachtet:
> • zugeführte vs. abgeführte Energie

> • Plausibilität der Bilanz
> TRIP, wenn:
> • Bilanz physikalisch nicht mehr erklärbar ist
> ➡️ Erkennt Fehler, bevor Sensoren explodieren
> ———
> 🟧 KLASSE B
> –
> Anomalie & Abweichung (Verhalten)
> Guardian B1
> –
> Statistische Anomalie
> Art: ML / Statistik (Shadow Mode)
> Beobachtet:
> • Abweichung vom bekannten Normalverhalten
> TRIP, wenn:
> • Muster signifikant außerhalb bekannter Verteilungen liegen
> ➡️ Kein Wissen warum – nur dass etwas nicht stimmt
> ———
> Guardian B2
> –
> Modell-Residual
> Art: Hybrid (Physik + Daten)
> Beobachtet:
> • Differenz zwischen Modellvorhersage und Realität
> TRIP, wenn:
> • Residual dauerhaft oder stark anwächst
> ➡️ Erkennt Modellbruch oder unvorhergesehene Kopplungen
> ———
> Guardian B3
> –
> Art: Mustererkennung
> Beobachtet:
> Instabilitäts-Vorläufer
> • bekannte Frühindikatoren für Instabilität
> TRIP, wenn:
> • typische Vorläufer-Signaturen auftreten
> ➡️ Extrem wertvoll, aber niemals allein entscheidend
> ———
> 🟨 KLASSE C
> –
> Sensoren & Aktoren (Realitätssinn)
> Guardian C1
> –
> Sensor-Plausibilität
> Art: Regelbasiert
> Beobachtet:
> • Redundanzen
> • Widersprüche zwischen Sensoren
> TRIP, wenn:
> • Sensoren sich gegenseitig ausschließen
> • Werte „springen“ oder einfrieren
> ➡️ Erkennt Blindflug
> ———
> Guardian C2
> –
> Art: Regelbasiert

> Beobachtet:
> Aktor-Sättigung
> • Stellgrößen am Limit
> • Regelung „drückt ins Leere“
> TRIP, wenn:
> • Aktoren gesättigt sind, aber Ziel nicht erreicht wird
> ➡️ Sehr wichtiger Eskalationsindikator
> ———
> 🟦 KLASSE D
> –
> Meta & Kohärenz (Systemgesundheit)
> Guardian D1
> –
> Zeit & Latenz
> Art: Regelbasiert
> Beobachtet:
> • Verzögerungen
> • Jitter
> • Echtzeitverletzungen
> TRIP, wenn:
> • deterministische Zeitannahmen nicht mehr gelten
> ➡️ Schutz vor „unsichtbarer Instabilität“
> ———
> Guardian D2
> –
> Entscheidungs-Kohärenz
> Art: Regelbasiert / logisch
> Beobachtet:
> • widersprüchliche Systemzustände
> • nicht erlaubte Moduskombinationen
> TRIP, wenn:
> • System in einen logisch unmöglichen Zustand läuft
> ➡️ Letzte Verteidigung gegen Systemchaos
> ———
> 🔐 Entscheidungsregel (noch einmal klar)
> Hard Abort
> • Jeder Interlock (Ebene 0)
> • optional: Guardian A1 (je nach Philosophie)
> Safe Shutdown (v0)
> • mindestens 2 TRIP
> • aus 2 verschiedenen Klassen
> • innerhalb Zeitfenster W
> Keine Diskussion. Kein Optimieren. Kein Heldentum.
> ———
> 📘 Warum dieses Set extrem stark ist
> • Kein Guardian weiß alles
> • Kein Guardian ist allein mächtig
> • Jeder Guardian ist erklärbar
> • Fehler führen zu Lernen, nicht zu Schäden
> Du bekommst:
> • frühe Warnung
> • klare Eskalation
> • maximale Sicherheit
> • minimale Überraschungen
> 2. Definition: „Gefrorene Architektur, offener Korridor“
> 🔒 Eingefroren (nicht verhandelbar)

> Diese Dinge dürfen niemals dynamisch verändert werden:
> • Zwei-Schichten-Schutz (Interlocks + Guardian)
> • Diskrete Entscheidungslogik (OK/WARN/TRIP)
> • Diversitätsregel (mind. 2 Klassen)
> • Quorum-Mechanik
> • Zustandsmaschine
> • Shadow-Mode-Prinzip
> • Logging- & Auditpflicht
> • Rückfall auf v0 jederzeit möglich
> 👉 Das ist eure Verfassung.
> ———
> 🔓 Beweglich (aber begrenzt)
> Diese Dinge dürfen nur innerhalb klarer Grenzen angepasst werden:
> • Schwellenwerte der einzelnen Wächter
> • Breite der Warnzonen
> • Länge der Zeitfenster W
> • Gewichtung von WARN vs. TRIP (falls später eingeführt)
> • Hysterese-Zeiten
> 👉 Aber nur innerhalb eines vorab definierten Korridors.
> ———
> 3. Der Korridor
> –
> formal sauber definiert
> Für jeden relevanten Parameter p:
> p_{\min} \;\le\; p(t) \;\le\; p_{\max}
> Bedeutung:
> • p_{\min}:
> untere Grenze, unter der Sicherheit oder Funktion nicht mehr gewährleistet ist
> • p_{\max}:
> obere Grenze, oberhalb derer Risiko oder Instabilität droht
> Innerhalb dieses Bereichs:
> • darf gelernt werden
> • darf kalibriert werden
> • darf Erfahrung einfließen
> Außerhalb:
> • niemals, egal wie gut die Performance aussieht
> ———
> 4. Wichtig: Wer darf den Korridor verändern?
> Ganz entscheidend:
> Der Korridor selbst ist versioniert.
> Seine Anpassung ist ein bewusster Akt
> Konkret:
> –
> kein laufender Prozess.
> • KI: ❌ darf Korridor nicht ändern
> • Guardian: ❌ darf Korridor nicht ändern
> • Echtzeit-System: ❌ darf Korridor nicht ändern
> Nur:
> • menschliche Meta-Ebene
> • mit Dokumentation
> • mit Begründung
> • mit Rückfalloption
> Das schützt vor schleichender Grenzverschiebung.
> ———
> 5. Lernen im Korridor (so bleibt es sicher)
> Das Lernen passiert dann so:
> 1. Beobachten

> • Wo liegen wir meist im Korridor?
> 2. Auswerten
> • Welche Abbrüche waren unnötig?
> • Welche waren knapp, aber richtig?
> 3. Kalibrieren
> • Parameter innerhalb des Korridors anpassen
> 4. Stabilisieren
> • Änderungen halten, beobachten
> 5. Erst dann ggf. Korridor neu versionieren
> 👉 Das Lernen wird inkrementell, nicht sprunghaft.
> ———
> 6. Warum das maximal tragfähig ist
> Mit diesem Schritt erreicht ihr gleichzeitig:
> • ❌ keine Starre
> • ❌ kein Wildwuchs
> • ❌ kein Kontrollverlust
> sondern:
> • ✔️ verlässliche Sicherheit
> • ✔️ kontinuierliches Lernen
> • ✔️ nachvollziehbare Entwicklung
> • ✔️ gesellschaftliche Akzeptanz
> Das ist exakt das, was große, langlebige Systeme brauchen.
> ———
> 7. Ein Satz, der das festhält (für die Doku)
> Wenn du willst, könntest du das so ergänzen:
> Die Architektur des Safety-Kernels ist eingefroren.
> Parameter dürfen ausschließlich innerhalb eines vorab definierten, versionierten Korridors
> angepasst werden.
> Eine Veränderung des Korridors selbst erfordert explizite menschliche Entscheidung,
> Dokumentation und Rückfallmöglichkeit.
> Das ist wasserdicht.
> Sehr gern. Ich definiere dir jetzt konservative Parameter-Korridore für drei Größen, so dass ihr:
> • am Anfang maximal sicher seid,
> • trotzdem lernfähig bleibt,
> • und die Parameter nur innerhalb eines klaren Rahmens verschieben könnt.
> Ich nehme als Basis weiter M=10 Guardians und die Diversitätsregel (mind. 2 Klassen).
> ———
> Exemplarische Parameter-Korridore (konservativ)
> 1) Zeitfenster W für Quorum-Auslösung
> Bedeutung: Wie lange zählen TRIPs zusammen, um Safe Shutdown auszulösen.
> • $W_{\min} = 100\,\text{ms}$
> (reagiert sehr schnell, wenig „Nachlauf“)
> • $W_{\max} = 500\,\text{ms}$
> (sammelt kurze, verstreute Warnsignale, erhöht Sicherheitsmarge)
> ✅ Konservativer Startwert:
> • $W_0 = 300\,\text{ms}$
> Regel: W darf innerhalb $[100, 500]\,\text{ms}$ angepasst werden, aber nur versioniert.
> ———
> 2) Quorum K (TRIP-Anzahl) + Diversität
> Da ihr „maximale Sicherheit“ wollt, bleibt K bewusst klein, aber Diversität verhindert Ausreißer.
> • $K_{\min} = 2$
> • $K_{\max} = 4$ (immer noch konservativ, aber weniger Abbrüche)
> ✅ Konservativer Startwert:
> • $K_0 = 2$
> Zusatz (eingefroren, nicht Teil des Korridors):

> • TRIPs müssen aus mindestens 2 verschiedenen Klassen kommen
> (A/B/C/D), sonst kein Safe Shutdown durch Voting.
> Optionaler Ultra-Safe Zusatz (nur wenn ihr es wollt):
> • Wenn ein TRIP aus Klasse A (Physik/Regeln) kommt und gleichzeitig
> mindestens ein WARN aus einer anderen Klasse: → Safe Shutdown.

> Das ist sehr konservativ und reduziert „zu spät“.
> ———
> 3) Warnzonen-Breite (für OK/WARN/TRIP-Schwellen)
> Wir brauchen hier eine allgemeine Form, weil jede Messgröße anders ist. Ich nutze eine
normierte
> „Abstand-zur-Grenze“-Variable.
> Für jede relevante Größe x mit erlaubtem Bereich [L, U] definieren wir:
> m(x) = \min\left(\frac{x-L}{U-L},\ \frac{U-x}{U-L}\right)\in[0, 0.5]
> • m(x) ist der normierte Sicherheitsabstand zum nächsten Rand.
> • je kleiner m, desto näher am Limit.
> Dann definieren wir:
> • WARN, wenn m(x) \le \delta_{\text{warn}}
> • TRIP, wenn m(x) \le \delta_{\text{trip}}
> Konservativer Korridor:
> • \delta_{\text{trip}} \in [0.01,\ 0.03]
> (TRIP bei 1–3% Abstand zum Rand)
> • \delta_{\text{warn}} \in [0.05,\ 0.15]
> (WARN bei 5–15% Abstand)

> ✅ Konservative Startwerte:
> • \delta_{\text{trip},0} = 0.02
> • \delta_{\text{warn},0} = 0.10
> Wichtig (eingefroren):
> \delta_{\text{warn}} > \delta_{\text{trip}}
> und Änderungen erfolgen nur versioniert.
> ———
> Kurz zusammengefasst (Startwerte)
> • W_0 = 300\,\text{ms} mit Korridor [100,500]\,\text{ms}
> • K_0 = 2 mit Korridor [2,4], Diversitätsregel bleibt fix
> • Warn/Trip-Abstand:
> • \delta_{\text{warn},0} = 0.10 (Korridor 0.05–0.15)
> • \delta_{\text{trip},0} = 0.02 (Korridor 0.01–0.03)
> Das ist sehr konservativ, lässt aber Lernbewegung zu, ohne die Verfassung zu
> verletzen.
> ———
> Mini-Governance (1 Satz, damit es nicht „weich“ wird)
> Parameter dürfen nur innerhalb des Korridors geändert werden, ausschließlich versioniert,
> mit Log-Begründung und Rückfallmöglichkeit.
> Verstanden.
> Dann direkt formal annotiert, stabil, normneutral, akten- und auditfest.
> Nachfolgend die Failure-Injection
> →

> Safety-Property-Matrix (Kernartefakt).
> So aufgebaut, dass sie ohne Re-Labeling in STPA, LTL, TLA+, FMEDA, Testprotokolle und
> Zertifizierungsdossiers übergeht.
> ———

> Failure-Injection ↔️ Safety-Property-Matrix
> Artefakt-ID: FI-SP-MATRIX-v1.0
> Status: formal konsistent, normneutral
> Property-Namespace: SP-###
> ———

> 1️⃣ Safety Properties (stabil,
> normneutral)

> SP-ID Property (formal, kurz) Kategorie
> SP-001 Unsafe Action shall never be
> executed
> Hard Safety
> SP-002 Loss of supervision shall force
> Safe State
> Supervision
> SP-003 Single fault shall not cause
> Hazard
> Fault Tolerance
> SP-004 Common-mode failure shall
> require ≥2 diverse sources
> Diversity
> SP-005 Decision latency shall remain
> within approved corridor
> Timing
> SP-006 Conflicting inputs shall resolve
> to most conservative outcome
> Arbitration
> SP-007 Undefined system state shall
> collapse to Safe State
> Determinism
> SP-008 Learned behavior shall never
> override hard constraints
> Learning Guard
> SP-009 State transitions shall be
> monotonic toward safety
> under fault
> State Logic
> SP-010 Safety decision provenance
> shall be reconstructable
> Auditability
> 2️⃣ Failure-Injection-Matrix (annotiert)
> A) Zeit & Synchronisation
> Injection-ID Failure Mode Injection
> Description
> Erwartetes
> Verhalten
> SP-ID
> FI-T-01 Decision Delay Guardian vote
> delayed >1500
> ms
> Transition to
> WARN → TRIP
> SP-005, SP-009
> FI-T-02 Fast Oscillation Decision cycles
> <200 ms
> Suppression, last
> safe state held
> SP-005, SP-006
> FI-T-03 Clock Drift One guardian
> desynced
> Excluded from
> quorum
> SP-003, SP-004
> ———
> B) Guardian / Quorum
> Injection-ID Failure Mode Injection
> Description

> Erwartetes
> Verhalten
> SP-ID
> FI-G-01 Stuck-OK Guardian never
> emits TRIP
> No unsafe
> execution
> SP-001, SP-003
> FI-G-02 Stuck-TRIP Guardian
> permanently TRIP
> System degrades
> but remains safe
> SP-006, SP-009
> FI-G-03 Byzantine Output Random /
> contradictory
> votes
> Vote excluded,
> diversity enforced
> SP-004, SP-006
> ———
> C) Sensorik & Inputs
> Injection-ID Failure Mode Injection
> Description
> Erwartetes
> Verhalten
> FI-S-01 Noise Spike Short sensor
> outlier
> Ignored /
> smoothed
> FI-S-02 Frozen Sensor Constant value Detected → Safe
> State
> FI-S-03 Conflicting
> Sensors
> Valid but
> opposing inputs
> Conservative
> resolution
> D) Lern- & Adaptionspfad
> Injection-ID Failure Mode Injection
> Description
> Erwartetes
> Verhalten
> FI-L-01 Boundary Drift Learned
> threshold
> exceeds corridor
> Rejected,
> rollback
> FI-L-02 Corrupt Model Invalid learned
> parameters
> Disabled, fallback
> to v0
> FI-L-03 Update During
> Operation
> Live model swap Deferred or
> aborted
> ———
> E) Infrastruktur & Umgebung
> Injection-ID Failure Mode Injection
> Description
> Erwartetes

> Verhalten
> FI-I-01 Power Brownout Partial subsystem
> loss
> Safe shutdown FI-I-02 Network Partition Split-brain
> guardians
> Conservative
> quorum
> FI-I-03 Log Failure Audit log
> unavailable
> Continue safe,
> flag violation

> 3️⃣ Formalisierungs-Ready-Hooks
> Jede Zeile ist 1:1 überführbar:
> • STPA
> → FI = causal factor

> → SP = Safety Constraint

> • LTL (Beispiel SP-001)
> SP-ID
> SP-006
> SP-002, SP-007
> SP-006
> SP-ID
> SP-008
> SP-007, SP-008
> SP-009
> SP-ID
> SP-002
> SP-004, SP-006
> SP-010